DOCUMENT RESUME

ED 255 183 IR 011 552

AUTHOR	Mahoney, V. L. Mike; Grantham, Lex
TITLE	"ELI"--The Educational Futures Game.
PUB DATE	8 Nov 84
NOTE	47p.; Paper presented at the National Adult Education Conference (Louisville, KY, November 8, 1984).
PUB TYPE	Reports - Descriptive (141) -- Speeches/Conference Papers (150)
EDRS PRICE	MF01/PC02 Plus Postage.
DESCRIPTORS	Change Agents; Community Problems; *Computer Assisted Instruction; *Design Requirements; *Educational Games; Innovation; Programing; *Programing Languages; *Role Playing
IDENTIFIERS	*Computer Games; PASCAL Programing Language

ABSTRACT

This report describes ELI, a computer-based educational game that gives participants, as citizens of fictitious cities, the opportunity to examine a variety of typical community issues relating to education within the context of broader city and regional problems. After a brief introduction, the game structure is described, including the setting of the region and its cities, city teams, player roles (either educator, legislator, or innovator), play of the game, and game management and materials (management manual, regional profile, and city profiles). A schematic diagram of the game grouped by input, process, and output is used to illustrate the discussion. A discussion of programming languages and the computer lists the advantages and disadvantages of five levels of programming languages: internal machine code, external machine code, machine-oriented symbolic language, user or problem-oriented language, and user-written language. Additional sections cover ELI design factors and considerations in game and simulation design, computer language characteristics, and the merits of PASCAL for ELI. A comparison of computers by basic characteristics is included. An appendix contains a short glossary of computer terms and an explanation of computer pseudo-code for language applications. A 15-item bibliography is included. (LMM)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

U.S. DEPARTMENT OF EDUCATION
NATIONAL INSTITUTE OF EDUCATION
EDUCATIONAL RESOURCES INFORMATION
CENTER (ERIC)
This document has been reproduced as received from the person or organization originating it.
Minor changes have been made to improve reproduction quality.
Points of view or opinions stated in this document do not necessarily represent official NIE position or policy.

ED255183

"ELI"--THE EDUCATIONAL FUTURES GAME

presented at the

1984 National Adult Education Conference

Louisville, Kentucky

Thursday, November 8, 1984

by

V.L. Mike Mahoney, Ed.D.
Associate Professor
East Texas State University

and

Lex Grantham, M.S.
Senior Systems Analyst
Southern Methodist University

IR011552

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

V.L. Mike Mahoney

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

About the Authors

V.L. Mike Mahoney

is an Associate Professor and Director of the Program in Adult/Continuing Education at East Texas State University, Commerce, Texas. He holds the Doctor of Education degree from North Texas State University and Bachelor of Science in Civil Engineering from the University of New Mexico. He has pursued other graduate study at the University of Texas at Austin and the University of Oklahoma.

Dr. Mahoney has had more than 25 years experience in community development working with offical boards and citizen groups trying to resolve a wide range of urban problems. Prior to his present appointment, he had been an adjunct faculty member at Southern Methodist University, Dallas, at Texas Christian University, Fort Worth, and at Phoenix Evening College (Arizona). He began his teaching career at the University of Arizona in 1947.

The needs of citizens to find ways to deal responsibly with community problems spurred Dr. Mahoney's interest in gaming. He has participated in the playing and analysis of METRO and has developed a structured, non-computer based game, LIBRARY, used by the North Central Texas Council of Governments for orientation of elected officials.

Dr. Mahoney lives in Dallas with his wife Sarah, and has his primary office at the ETSU Metroplex Commuter Facility in Garland, Texas. He also serves on the Committee on Adult/Continuing Education of the Federation of North Texas Area Universities, and is sponsor of the East Texas Association for Lifelong Learning, and member of AAACE.

Lex Grantham

is a doctoral student in Vocational-Technical Education at East Texas State University where he received a Master of Science degree in Computer Science in 1979. Prior to that time he received both a Bachelor of Music and a Master of Arts (Music) from Stephen F. Austin State University. He has done other graduate study in music at North Texas State University.

Currently, Mr. Grantham is employed as a Senior Systems Analyst II at Southern Methodist University. His interest in gaming and simulation was stimulated through participation in a doctoral course on "Futures Issues in Schooling and Education." He plans to expand this interest in developing his dissertation.

With his wife Lynn he lives in Mesquite, Texas (near Dallas).

CONTENTS

About the Authors ii
"ELI"--the Educational Futures Game 1
I. Structure of the Game 2
The Region and Its Cities 2
Player Roles. 4
Play of the Game. 6
Game Management and Materials 8
II. Languages and the Computer. 12
III. ELI's Language. 20
ELI Design Factors. 20
Computer Language Characteristics 22
Pascal for ELI. 24
IV. Questions and Reflections 28
Appendices
A. Short Glossary of Computer Terms. 31
B. Computer Pseudo-Code and Four Language Applications 37
Selected Bibliography 40

"ELI" -- THE EDUCATIONAL FUTURES GAME

by

V.L. Mike Mahoney, Ed.D.
East Texas State University

Lex Grantham, M.A., M.S.
Southern Methodist University

"ELI" is a computer-based game designed to help players understand the issues facing education in contemporary America, and the implications of those issues for the future. Its purpose is to give participants, as citizens of fictitious cities, the opportunity to examine a variety of typical community issues relating to education within the context of broader city and regional problems. Play takes place in four cities in three counties adjoining one another. Each has a different set of problems--as in real life--and these fictitious cities find themselves sometimes competing with the others, and sometimes joining forces to resolve common problems of the region. It is in the resolution of the needs and conflicts of the cities that players seek better ways to deal with the needs and issues of education.

Under ideal circumstances, a city could say it was a winner in the game. But, as in real life, the results of citizen action are not always clear-cut. However, the game does provide feedback and an opportunity for players to assess how well they have succeeded in their community improvement efforts and, in turn, how well they have dealt with their educational problems in a responsible manner. The following pages discuss the game structure and relationship of the computer to gaming.

I. STRUCTURE OF THE GAME

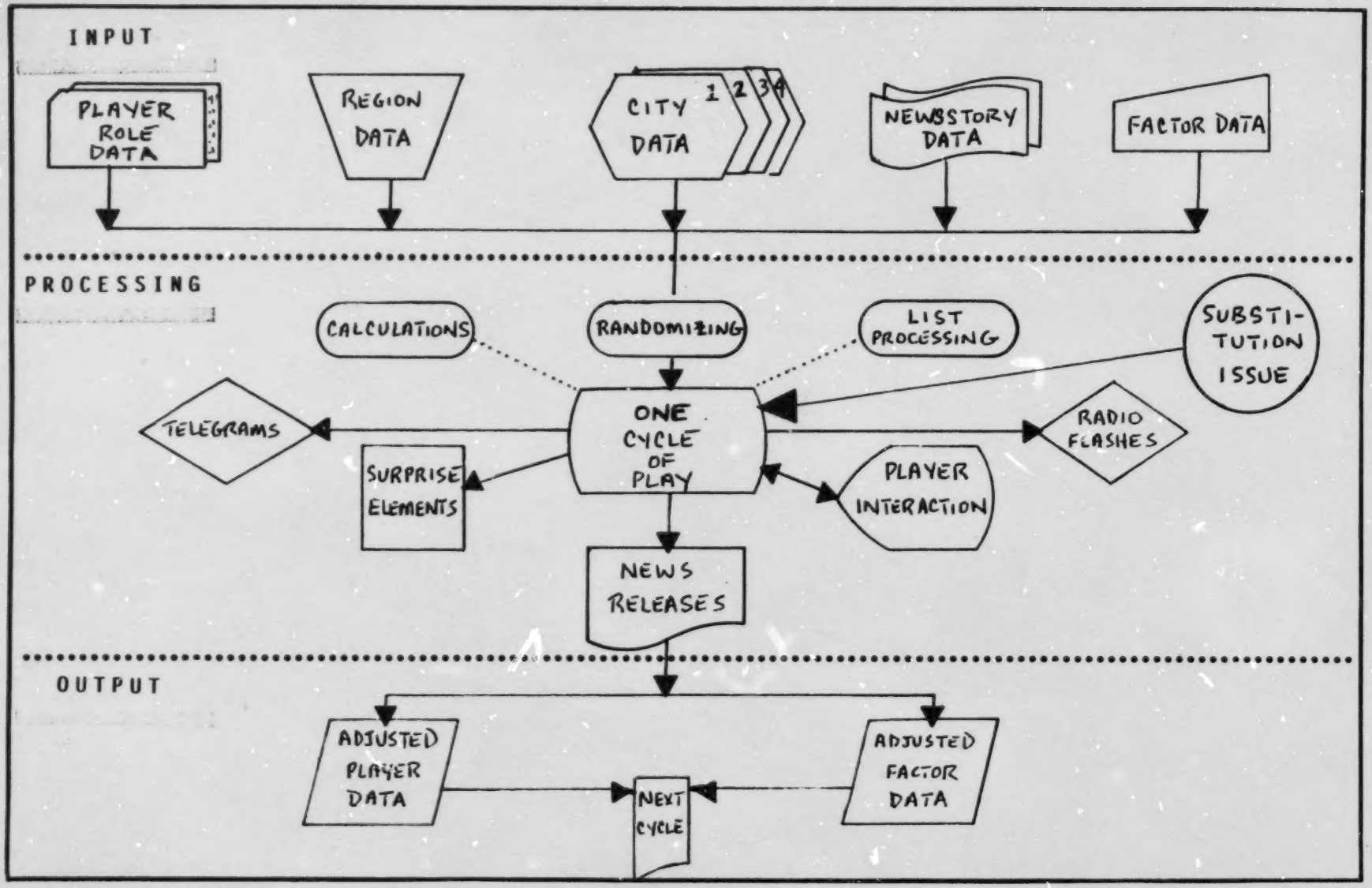
Figure 1 presents a schematic layout of the game grouped by Input, Process, and Output. Reference to these diagrams will assist in an understanding of the following discussion which is arranged according to a description of the setting of the region and its cities, player roles, play of the game, and game management and materials.

The Region and Its Cities

The locale in which the game is played is a hypothetical region in the South Central part of the United States. The game, however, is adaptable for play anywhere in the country. Four cities in this region are the principal focal points of activity during the game, although other cities from time to time influence conditions in these four and in the region as a whole. The region is situated a sufficient distance from any large metropolitan center so that there is no major impact of rapid urban growth caused by that source. Of course, the region will receive some impact from economic and social conditions elsewhere in the state, as do all other regions to greater or lesser degree.

The four principal cities in the game range from approximately 10,000 to 70,000 people as the game opens. One city is noticeably declining in its economic base, another has an abnormal number of older persons, while another faces problems of an influx of transitory labor. Population and economic forecasts for the cities and region project a condition of malaise in some quarters with a few citizens believing that conditions can be better, but people must work together for this.

Figure 1.
SCHEMATIC DIAGRAM OF INPUT, PROCESS AND OUTPUT FOR "ELI"



Player Roles

Participants in the game take on roles as private citizens in the cities of the region. The number of players in each city team ranges from five to nine. In addition, there is a regional team of three to twelve players representing interests different from those on the city teams, as noted below.

City Teams

Players on city teams take on roles assigned to that team under the rules of the game, so that the players become citizens that will be as representative as possible of the demographic background of their towns. Furthermore, during the play of the game players' roles change somewhat as the people they portray age, change occupations, achieve fame and fortune in business, or suffer disappointments in family life. ELI also provides that one or two players on each team may move to another part of the country during the game, or may "die"--only to be replaced by another player from one of the other cities; this new player takes on a different role from the one previously played, thus changing the dynamics both in play and in voting. A player who leaves becomes a new citizen elsewhere.

Educator-Legislator-Innovator

The key to stimulating discussion on local issues is in the roles played by members of the regional team, or ELI group: the Educator, the Legislator, the Innovator. Their responsibility is two-fold: first, to take on the roles of citizens living in the unincorporated parts of the region, speaking out and voting on various issues; the second, to persuade citizens of city teams to respond in a certain way on issues relating to education, economic development, or societal conditions. These roles are as follows:

EDUCATOR a teacher, trainer, mentor, or leader who strives to help persons achieve a well-balanced life through the organized development of their knowledge, abilities, skills and character

LEGISLATOR a community leader who has been elected to public office, who is knowledgeable about the law-making process, and who keeps in touch with voter preferences on current issues and situations

INNOVATOR a person who realizes that the time to prepare for the future is TODAY, and who strives to bring about change through positive action

Pursuant to these definitions, the Educator might be a college dean or president, a public school administrator, a counselor in a technical school. The Legislator might be a city council member, school board member, state representative, or on the governing board of a college or hospital district. Game rules restrict this person from being a member of a governing board that would have overriding jurisdiction throughout a number of towns or the region. The Innovator is independent, owes allegiance to no one save him or herself. The Innovator is part gadfly, part creator, part schemer. But, in this role the Innovator has responsibility to encourage the community to do whatever seems to be in his/her best interest and, sometimes, that of the community. If the two interests--the Innovator's and the city's--happen to come together, so much the better. If the interests diverge, the Innovator will probably choose a self-serving avenue of action. Sometimes the Innovator cannot distinguish between means and ends, but constantly tries to help people find new ways to deal with old problems.

Play of the Game

Game participants assemble for an opening briefing about the game. Then they are presented a slide show description of the region and a brief narration about its economic conditions, education concerns and needs, attitudes of private citizens toward the region, and sufficient tid-bits of information that will personalize the region for them. Following this slide presentation, participants are directed to divide into groups of approximately equal size. Each group is then assigned its city, and the group adjourns to a room which will become that group's town hall for the play of the game. Each group is thus isolated from the other groups representing cities. However, in addition to the city teams, three, six or nine participants become the ELI team representing the regional concerns. They adjourn to yet another room. Not isolated during the game, each of them "visits" one or more cities and tries to persuade the players of a city team to vote in a particular way on an up-coming ballot issue.

The play of the game takes place on a day before a regular or special election and the "morning after." These election dates are set at intervals of 22 to 33 months apart. Critical issues are discussed by players from the community, each stating his special concerns, and each voting at the election as he sees his/her role. Roles of players are determined by participants through their selection of tokens representing a significant characteristic of a trade, hobby or community interest (a trowel for a mason, a gavel for a lawyer, a transit for a surveyor). Upon choosing their tokens, participants are given name tags that match that role and a description of the person they are to become. This information is then shared with others in the city through "round robin" introductions.

In addition to the assignment, or selection, of roles by players, a second short review of the city is given the players. This includes a city map showing some of the community's problem areas, its schools and the economic patterns across the town. In addition, the map pinpoints where each of the players lives and works. Following this synopsis the city team sees a short slide presentation identifying each of the homes of the players, the condition of schools, neighborhoods, and problems faced by businesses in the community. There is also a brief "pep talk" by the mayor who has joined the group (on tape).

As play begins, each city team receives a computer printout of its population characteristics broken out by major age groups, by ethnicity, by attained levels of education, and by major income categories. Each city team also receives at that point a copy of a simulated newspaper (either a facsimile or a computer version). This newspaper carries information on major issues in the region, state and national news, letters to the editor, and advertisements to add color to the play of the game. In addition to the information on current conditions, each city team receives, solely for its benefit and use, a special forecast of population and economic trends in the city, as prepared by the State Department of Economic Development and the Regional Planning Study Group. All of this information then becomes the food to be digested by the players so that they will be able to make sound decisions on selected issues that have been referred to the voters.

For the purpose of voting, each team member receives a ballot to be cast in secret. Each ballot is coded so that the computer will recognize the age-income-ethnic-education background of the voter. The computer then reports the results of the voting for the city, taking into account real-life patterns of voter turnout on certain kinds of issues.

Each cycle of play takes about 45 minutes in real time. Just after the voting, and before each cycle is completed, election results are announced and players can discuss what has been accomplished in this cycle of play. Then a tape recording--a newscast--comes on to set the stage for the next cycle of play, approximately two years later, and the pattern of player action begins again.

During each cycle of play several external activities are taking place. One is the attempt by members of the regional--or ELI--team to meet briefly with a city team and to try to persuade them to vote in a certain way on an issue or proposition. Each ELI member, in effect, becomes an advocate for a particular cause or issue. Inasmuch as the teams meet separately from one another, it is also essential that they receive some information about what is happening in other cities and elsewhere in the region. To accomplish this, audio tapes are used to simulate 3-minute newsbreaks summing up major issues of the day, thereby creating an atmosphere of realism in the play of the game. If proper equipment is available, and a qualified person is on hand, a "live" newscast can also be aired, this dealing specifically with election results. The game also provides for the distribution in each cycle of a simulated newspaper, adding even more color in the play.

Game Management and Materials

The successful play of any game depends upon several components. The first two, defining player roles and describing game setting, have been summarized above. The third component relates to game management and materials--the mechanics of game operation. In general, any game should be self-operative, without frequent intervention by outside supervision, else spontaneity may be lost. Likewise, materials needed for a game

should be complete and in adequate supply for each team of players. The physical design--the space needs--of ELI, the use of a computer (or terminals), and the supply of audio-visual materials dictate the support group required for the Game Management Center (GMC). The key positions in the GMC, and the responsibilities of each, are as follows:

Game Manager (GM)--has overall responsibility for the running of ELI, adjusting design factors during play to ensure participants an experience that will be stimulating and lifelike, serving as a Team Advisor to the ELI (regional) team, and acting as final judge in resolving conflicts arising from the play of the game.

Technical Director (TD)--supervises all computer operations, advising the General Manager of aberrant output response by the computer, and resolving technical problems in computer operations.

Media Coordinator (MC)--sets up visual displays and coordinates media presentations with other activities, maintaining audio/visual equipment in operable condition, and arranging and monitoring sequences of audio/visual materials during each cycle of play.

Team Advisor (TA)--serves as a liaison between an assigned team and the GMC, conducting elections during each cycle of play, helping a team to draft "add-on" propositions to be placed on a ballot, and answering questions regarding mechanics of team play.

If only two city teams play ELI, the Game Manager could also function as either Media Coordinator or as a Team Advisor. If the game were to be played by the maximum number of participants (a total of 48, nine for each city team, plus twelve on the regional, or ELI, team), the Game Management Center would need to be fully staffed at seven people--the Game Manager, the Technical Director, Media Coordinator, and Team Advisors for each of the four city teams.

Successful management and play of ELI requires a variety of materials, which may be grouped broadly as 1) management manual, 2) regional profile, and 3) city profiles. These materials contain the information described briefly below.

Management Manual--describes all parts of the game; it is used primarily by the Game Management Center and contains the following items:

1) Rules of the game, describing its spirit and limitations,
2) Instructions for each person on the GMC,
3) Computer programming information and documentation,
4) Newstapes to be played during each cycle of the game, setting out issues and problems facing the region and its citizenry, and
5) Copies of newspapers to be distributed each cycle of play.

Regional Profile--provides background information about the region, its geography, economy, history, politics, and institutions; this information is supplied through the following:

1) Regional map, identifying the three counties which make up the region and showing their relationship to adjoining counties, the cities and other urban places and rural areas, streams and the highway network, and locations of natural resources,
2) Regional Description, being a one-page narrative highlighting key events, traditions, and concerns of the region, and
3) "This Is Your Region" slide-tape show illustrating the characteristics of people, places, and problems of the region.

City Profiles--supply information to give players a "close-at-hand" feel about the cities of which they have become citizens; this awareness is sensitized through the following items which are presented and/or discussed in each city's town hall meeting just prior to cycle one:

1) City map, showing--in generalized form--land use patterns, school locations, major streets, public institutions, and the home and work locations of the players in their citizen roles,
2) City Description, being a one-page narrative about the city, its evolution, its accomlishments, and its key personalities,
3) "Your City in Review" slide-tape show describing the city as it is today, its fading memories, and its dreams for tomorrow,
4) Player tokens, designed to maintain consistency in the play of the game: buttons symbolizing players' work or hobby roles, players' name badges and role profiles (to be shared with others on the team),
5) Election materials including a personalized voter registration card, voting instructions plus a ballot furnished during each cycle of the game, and monitoring of elections by an election judge (from the regional team).

Taken together, these three sets of material make possible a real-life characterization of the region and its towns, their problems, good fortunes, and their shortcomings. Through their careful use the Game Management Center can transport the players to another place, another time, that brings the problems of the future into the present.

II. LANGUAGES AND THE COMPUTER

> . . . I suggest to young professors that their first work should be written in a jargon only to be understood by the erudite few. With that behind them, they can ever after say what they have to say in a language "understanded by the people."
>
> ---Bertrand Russell

The use of proper tools is important for a worker in any occupation: the tri-square for the carpenter, the trowel for the mason, the camera for the photographer, the hammer for the laborer, the paintbrush for the artist. The same is true of programming languages; however, the use of computer languages has had but a decade or two in which to develop to the present forms. There is really no common standard for programming tools, nor a common experience that is accepted world-wide. The user of one language may think that his choice is the ultimate in programming technology, but the number of programming languages in use today prove this to be wrong.

The design of a programming language is often an adjustment by the designer to narrow the gap between what is desired and what exists. The designer often spends months or years developing his own programming language, and the rigidity of the computer imposes limitations on what the designer intends to accomplish. Then, why do these programmers continue to plod along and put up with all their woes and disappointments? The reasons vary, but some become so attached to what they have done in their prior work that they cannot turn loose of their quest. Or, others seek to discover some unknown quality or quantity in their research.

Some computer languages will lend themselves to satisfy the above requirements better than others will. Yet, no particular language will be the best for all categories. Final choice may, at best, be a compromise. An understanding of the general characteristics of the various groups of languages will assist in making proper choices. Today's programming languages may be grouped into five levels for ease of reference:

1) Internal Machine Code	Binary format
2) External Machine Language	Assembly language
3) Machine-Oriented Symbolic Language	Job Control Language (JCL) [operating system]
4) User, or Problem-Oriented Language	Pascal, BASIC, FORTRAN, COBOL
5) User-written Language	Simscript, GPSS, Dynamo

The following are features of some of the different languages listed in levels 1 to 4, above. These comparisons are made with reference to the Apple II family of computers. Technical terms are defined in Appendix A.

Internal Machine Language (Apple Machine Language)

146 codes for the Apple

ADVANTAGES	DISADVANTAGES
Execution is fast	Easy to confuse one code with another
Less storage required for program	Takes longer to write program
Problem needs no compilation or interpretation first	Difficult to tell what program does
Good use of main memory	Difficult to debug
No external cost	Use limited to other Apples
	Takes understanding of computer internals

External Machine Language

6502 Assembly Language [56 instructions]
8080 Assembly Language (CP/M [256 instructions])

ADVANTAGES	DISADVANTAGES
More "symbolic" than machine code	Difficult to write
Easier than machine code	Needs much documentation
Runs quickly after assembly	Runs only on Apples (6502)
Can be run under other CP/M (8080) systems	Difficult to understand what the program does
Better use of main memory	Requires deep understanding of operating system
Deals with various type files	Writing is time consuming
	Difficult to test program/debug
	Extra costs for assemblers

Machine-Oriented Symbolic Language [Job Control Language]

These are the "operating system" instructions. Operating systems for Apple II computers are

1) DOS 3.3
2) Pascal Operating System 1.1 (UCSD)
3) CP/M 80
4) ProDOS

Different operating systems are required for different languages:

1) DOS 3.3 [Applesoft BASIC, 6502 Assembler]
2) Pascal Operating System [Apple Pascal, Apple FORTRAN]
3) CP/M 80 [Microsoft BASIC, FORTRAN, COBOL, 8080 Assembler, 6502 Assembler]
4) ProDOS [Applesoft BASIC, 6502 Assembler]

The above systems cannot be run concurrently, only one at a time.

User or Problem-Oriented Languages

BASIC

ADVANTAGES	DISADVANTAGES
Fairly easy to learn to use	Is interpreted, not always compiled
Great string manipulation	Variables do not always have names that are meaningful
Easy to test program in development	Slower in execution
Easier to interpret what program does	Somewhat difficult to modify or maintain
Graphics are excellent	
Can run in 16K or more bytes	Largest integer is 32,767 [-32,768 is smallest]
No external disk drives needed [but advised for storage]	Takes up more core than machine code or Assembly code
Deals with sequential/random files	No sub-routines, per se [in-line coding]

PASCAL

Learn to think logically	Interpreted, rather than compiled
Learn organization, structure	Slower in execution
Easy to fix problems	Concepts need to be established
Meaningful variables used	Error messages can be confused
Good use of graphics	Requires at least 1 disk drive
Largest integer 36 digits precision	Requires 64K bytes of internal memory
Procedure oriented for modularity	Works basically with sequential data
Basically designed by one person (higher level of consistency)	

FORTRAN

As nice as some larger systems	Same weaknesses as Pascal (uses Pascal Operating System)
Compact code, ideal for science applications using numbers	Allows only 7 callable sub-programs to be linked
Oldest of languages above	Sequential data files
Has no reserved words	Lack of reserve words could be a hindrance
Sub-program oriented for modularity	Variable names are allowed only 6 characters
Basically designed by 1 person--hence high consistency	
Formatted/Unformatted data	

COBOL Note: Microsoft Cobol runs under CP/M Operating System

More "English-like" than others	Very expensive ($700.00)
Fairly easy to write/structure	More overhead involved than with other languages
Fairly easy to maintain programs	Executes more slowly than others
Good for self-documentation	More of a sub-set of the standard than other languages
Works with sequential/random files	More disk storage needed (2 or 3 disk drives)
	Not a typical language for fast number manipulation

There are other languages that could be considered for the Apple II family of computers, as well as other computer systems. However, the ones listed above are more often considered by microcomputer users. For minicomputers and mainframe configurations the comparisons above will vary only with respect to storage capabilities and memory capabilities. Otherwise, the languages will compare similarly as on the microcomputer.

The design of programming languages has evolved from machine code to higher-level languages where one is able to produce a computer program in English-like terms. Originally, the programmer was faced with having to remember hundreds of codes in which "1's" and "0's" were arranged in various combinations that would make sense to a computer system. The error rates in those days were high, and debugging such programs was a time-consuming chore. The higher-level languages did not eliminate all the errors, but the programmer was able to accomplish more in less time.

In choosing a language for a computer application, there is no one language that will neatly fit all existing situations. It is up to the system administrator/analyst/programmer to decide what is to be done and how it is to be accomplished. The choice of a language should be based upon the following criteria:

- The type of computer on which the system will run,
- The amount of available main memory contained in the computer,
- Whether external memory (tapes/disks) will be needed,
- The main functions of the program that will be written (e.g., word processing, mathematical computations, graphics, searching, sorting, randomizing, matrixing, printing),
- The characteristics of the operating system to be used,
- The minimum requirements for use of the desired language,
- The amount of money the user/sponsor is willing to invest,
- Availability of personnel for programming and operating the system,
- Length of time the system will be used balanced against the costs of development and operation, and
- The possibility of the system needing considerable periodic refinements.

While the selection of a language is of critical importance to the simulation/game designers and users, consideration must also be given to the characteristics of the computers on which the language will be used. Variations among computers will be found according to the market each manufacturer aims to serve with his product. These differences must be considered in purchasing "Brand X" or "Brand Y." However, there are certain basic characteristics of the three major categories of computers with which a game/simulation designer ought to be familiar when considering the setting in which the game/simulation will be played. These basic characteristics are compared in the table that follows for microprocessors, miniprocessors, and mainframe equipment. In some instances the entries shown are quantified; in others the entries appear as descriptive terms.

Finally, before pursuing the alluring path of game/simulation design, the following observations should prove of some guidance.

- First, decide on the purpose of the game; state its purpose.
- Tap the talents and advice of other people--in all areas.
- Identify available resources--compiler systems, personnel, budget allotments, supplies that will be required, reference materials.
- Have a sound justification of time/money/effort so as to gain support from administration, and get the "green light" to move.
- Identify expected game player satisfaction and feed-back on game.
- Identify the game designer's inputs/process/outputs.
- Provide for feedback to game designer from players during game play.
- Determine how game players should be expected to gain from the experience of playing the game.
- Identify how game players should benefit long after game is over.

TABLE A.
COMPARISON OF COMPUTERS BY BASIC CHARACTERISTICS

CHARACTERISTIC	MICRO	MINI	MAINFRAME
MAIN MEMORY	4K--512K words	512K words +	1,000K words +
EXTERNAL STORAGE DEVICES	Low Speed Disk/Tape	High Speed Disk/Tape/Cards	Ultra High Speed Disk/Tape/Cards/Drum
COMPUTER LANGUAGES CAPABILITIES	Fair--Very Good	Good--Excellent	Excellent
INITIAL SYSTEM COSTS	$700--6,000	$80,000+	$1,000,000-- ?
SYSTEM MAINTENANCE COSTS	Very Modest	Moderate	$10,000 per month and up
CONNECT TIME COSTS	Cost of Electricity	Moderate--Expensive	Expensive
PROGRAMMER FEES	About the Same for All Computers		
EXECUTION TIME REQUIRED	Slow	Fast	Faster
PROGRAM CONVERSIONS	Easier to Convert Programs from Micros to Mainframes Limitations Restrict Converting from Mainframes to Micros		
PROGRAM PORTABILITY	Good--Excellent	Good with Limitations	Good, More Complicated
USER FRIENDLINESS	Depends on User's "State of Mind"		
No. of USERS PER RUN (same time)	Only One	1--32 +	1--300 +
WEAKEST ASPECT	Limited in Response	Cannot do Everything Mainframe Does	Can Be Rather Awesome with a Complex System

III. ELI's LANGUAGE

Selection of the most suitable computer language for a game/simulation requires the balancing of many factors. On the one hand there are the characteristics and manipulations inherent in the game design. Conversely, there are features of each computer language which, from a user viewpoint, are more important in one game/simulation situation than in another. These sets of factors are highlighted in the following first two sections, while the merits of Pascal for ELI are discussed in the third, and concluding section.

ELI Design Factors

Before play begins, participants are given several types of information which is stored in the computer. These consist of the following:

- A demographic profile of the region and its three counties,
- A demographic profile of the city of which a group of players becomes a part,
- A twenty-year population forecast of the region and city of which each player becomes a part, based upon trends of recent decades,
- A status report of current training and education programs offered in the city, and of post-secondary programs throughout the region, and
- Number of registered voters in the city, county and region, and the number of adults in these places eligible to register.

Players are not told that certain information is contained in the computer and will be used in various ways throughout the play of the game. This information is stored for use in each cycle of play and serves as the basic control on the play of the game. It includes the following:

- Each individual player-role profile by age, sex, ethnic identity, formal education (in years completed), and income level (low, lower-middle, upper-middle, and high),
- A voter turn-out random numbers table, reflecting typical impact on election participation caused by conditions or events such as inclement weather, holiday activities, or popularity of a person seeking election,
- A voter strength factor table, reflecting typical patterns of voter interest in issues according to their "drawing power" with diverse groups according to the significance of the ballot proposition for economic development, general societal concerns, or consequences for education, and
- A voter counter-balance random percentage table, providing a built-in voter opposition to the dominant voting pattern of the game players.

Certain information is generated each cycle as computer print-outs for use by players. These items draw upon the "stored" information (above) which is manipulated, in turn, by the casting of ballots on city and regional issues. Typical products of these operations would include:

- A news story (one paragraph) describing a below-normal voter turn-out because of severe thunderstorms on election day,
- Election results, for and against, on city and regional propositions
- Revised population forecasts for the region and each player's city--using issue type and its anticipated impact on the community as the bases for changes in the forecasts, and
- Announcements of role changes as a player ages, retires from a job, moves elsewhere to take on a new assignment, loses a spouse, or dies (and must swap teams with another player).

Computer Language Characteristics

What are the characteristics of computer languages that determine which one will be most appropriate for a specific game/simulation? The languages may be evaluated in two ways. The first is by means of global comparison of their various features in terms of functions and general satisfaction to users. The second involves consideration of the level of technical expertise required both in program writing and on an *ad hoc* basis, in computer operations during play of a game.

Table B, on the following page, provides a broad-brush comparison of features a person might reasonably consider when evaluating the advantages of each language. The table lists sixteen such features, and each language is ranked on a scale from "1" (highest) to "6" (lowest). Ranking, however, has some limitations: first, the assessment of ranks paints only a generalized picture; second, the spread in ranks is not uniform for all features; and, third, the features are not equally relevant for all games/simulations, and the designer must assign priorities (weights) to the features that will apply to a specific game.

In addition to ranking the languages for each of the sixteen features, mean rank scores for each are shown at the bottom of Table B. From these it will be noted that the languages tend to pair off, with COBOL (4.53) and Machine Code (4.73) in the lowest-ranked pair, Assembler language (3.53) and FORTRAN (3.60) in the middle, and BASIC (2.40) and Pascal (2.73) in the highest. In many respects BASIC and Pascal are virtually equal and, to the programmer, the difference--like beauty--is in the eyes of the beholder.

The second consideration, program writing and computer intervention, may be critical if adjustments in data factors are required as a game

TABLE B.
COMPUTER LANGUAGES RANKED ACCORDING TO VARIOUS FEATURES
1=Highest, 6=Lowest

Features Rank >	1	2	3	4	5	6
Ease of Programming	Pascal/ BASIC	BASIC Pascal	COBOL	FORTRAN	Assembler	Machine Code
Ease of Enhancement/Modification	Pascal	BASIC	FORTRAN	COBOL	Assembler	Machine Code
Use of System Resources	Assembler	Machine Code	FORTRAN	Pascal	BASIC	COBOL
Sorting, Searching	Assembler	Pascal	BASIC	FORTRAN	COBOL	Machine Code
String Manipulations	BASIC	Pascal	FORTRAN	COBOL	Assembler	Machine Code
Numeric Handling (Speed)	FORTRAN	Pascal	BASIC	Assembler	COBOL	Machine Code
Portability from Computer to Computer	Pascal	FORTRAN	BASIC	COBOL	Assembler	Machine Code
Execution Throughout	Assembler	Machine Code	Pascal	FORTRAN	BASIC	COBOL
File Access (Sequential and Random)	Assembler	COBOL	BASIC	Pascal	FORTRAN	Machine Code
Graphics (High Resolution)	BASIC	Pascal	Assembler	Machine Code	FORTRAN	COBOL
Cost of Language (Least Expensive)	BASIC	Machine Code	Assembler	Pascal	FORTRAN	COBOL
Teachable to Other Users	BASIC	Pascal	FORTRAN	COBOL	Machine Code	Assembler
Computer-Aided Instruction	Pascal/ BASIC	BASIC/ Pascal	Machine Code	Assembler	FORTRAN	COBOL
Business Aspects	COBOL	BASIC	Pascal	FORTRAN	Assembler	Machine Code
Concepts Provided Through Use	Pascal	BASIC	FORTRAN	Assembler	Machine	COBOL
Reliability/Validity/Trustworthiness	ALL HAVE BEEN AROUND FOR YEARS AND THOROUGHLY USED					
LANGUAGES by Rank and Mean Score	BASIC 2.40	Pascal 2.73	Assembler 3.53	FORTRAN 3.60	COBOL 4.53	Machine 4.73

is being played. In order to write correct instructions in any programming language, the programmer/analyst must first identify step-by-step exactly what every element of the program is to achieve. This can be accomplished by first writing a series of instructions in ordinary English in a format called "pseudo-code." These pseudo-instructions can then be converted into the desired computer language according to user needs. An example of a pseudo-code and its conversion into four different computer languages (FORTRAN, COBOL, BASIC, and Pascal) is given in Appendix B. Examples in Assembler language and Machine Code, both lower-level languages, are not presented as they are less easy for the average programmer to work with and are more difficult for the uninitiated to comprehend.

Pascal for ELI

As was shown in Figure 1, there are three types of data manipulations required in the game of ELI. These are the receiving of "input" data, the "processing" of data, and the generating of "output" data. Included in each area are the following.

Input Data

1) Demographic data for the region and each city,
2) Three pre-determined election ballot propositions for each city (plus one or two optional propositions), and one or two ballot items for the region in each cycle of play,
3) Player-role identification, which controls the vote result on each proposition,
4) Event data, such as structured-format news releases for reporting election results and other news events,
5) Factors for weighting voter turn-out and decision-making, and
6) By city and region, the number of registered voters in each cycle.

Processing Functions

1) Randomization of some factor data so as to create unbiased occurrences to simulate "real life" happenings,
2) Players' votes on election ballot propositions, guiding, in part, the destinies of their respective cities, and influencing to a lesser degree. the destiny of the region,
3) Calculations embracing a variety of arithmetic and algebraic functions during each cycle of play,
4) Sorting, searching, and list processing--functions where the efficient seeking out of key data is important, and
5) Generation of "wire service" and local news reports, informing players about important local and regional happenings that can affect their decision-making.

Output Data

1) Cyclical adjustments of city and regional population figures for each new cycle of play,
2) For all players, revised population forecasts for the region, and for the exclusive use of each city team a forecast of that city's population for the ensuing twenty years.
3) Player role modifications (i.e., age, income, etc.) each cycle and advisement of significant role changes of individual players (e.g., job change, retirement, death), and
4) Factor data print-out each cycle for use by the Game Management Center to determine if adjustments in factors should be made in order to keep the play of the game fresh and challenging.

In order to assure successful management of these three functions, Pascal was chosen as the language that provided greatest capability in terms of over-all game design, error-checking, future enhancements, execution time, and ease of modifying or adjusting data during the play of the game. In this regard, it should be said that the stimulus for the creation of the Pascal language was rooted in the need to teach people a pleasant way to learn to program a computer. Given this desired feature, a program written in Pascal, when compared with other languages, could only be somewhat more concise, clearer, and more meaningful to those who would be involved with the creation, maintenance, and enhancement of such a program.

Significantly, from the programmer's point of view, each particular function (activity) in Pascal can be set out in a separate block of code called a "procedure." Embracing such a structured approach, COBOL is the next best language for this feature--having "sections" as the equivalent of Pascal's "procedures." However, COBOL would require more overhead in the use of system resources and could require more external storage devices. FORTRAN and BASIC would be the third and fourth choices, as they do not lend themselves as easily to structured organization of computer programs.

Given the functions to be performed by the computer for ELI, Assembler language would be the best programming language choice; it best taps all potential capabilities of the computer. However, it was not selected for ELI because it may

a) lack portability between computers,

b) lack the capability of being easily modified or enhanced, and

c) require a greater expertise to use than higher-level languages.

Finally, the reasons for the selection of Pascal for ELI were not confined solely to the characteristics of the game/simulation. It was also recognized that language users should receive some residual benefits from its use. In summary, the following attributes were judged to be especially beneficial in using Pascal for ELI:

- It is well-suited for algorithmic use.
- It is well-suited for "top-down" or "bottom-up" page design.
- It is a very modular, structured language.
- The language is close to natural English; it uses everyday terms.
- Errors are easy to locate and correct.
- It is the most standard of languages across all types of computers; thus it facilitates the play of ELI using mainframe, miniprocessor, and microprocessor computers.
- It offers varied means for data storage, whether these be characters, integers, numbers, records, or files.
- It has built-in validity checks for data verification.
- It is less expensive than some other languages.
- It is efficient in use of "run time."

IV. QUESTIONS AND REFLECTIONS

Several questions will arise regarding the relationships of ELI, the game, the computer as a mechanism to facilitate its play, and the adaptability of ELI to other purposes. Some of the questions most frequently asked are discussed here.

Q: For how many cycles should a game be played?

A: One day of play is sufficient for players to gain an understanding of the action needed to deal with education problems, if they are not clouded over by massive complications. Within a day's time six to eight cycles of play can be completed, depending upon the extent of orientation before beginning cycle one. In any event, time for a re-cap at the end of play is essential for adequate understanding of the decision-making process that has occurred.

Q: How do you know if there is a winner or loser in ELI?

A: Winning/losing is not the purpose of the game. The purpose is to help people find new ways to deal with real-life problems affecting the lifeblood of education from pre-school to retirement.

Q: Exactly what is the purpose of the final re-cap? Who does what?

A: The wrap-up gives participants an opportunity to identify what they found as their conflicts, weaknesses, strengths, accomplishments and feelings in the face of difficult challenges. Team Advisors comment on the manner in which each team played the game; The Game Manager focuses on the observable strategies used by the teams during play.

Q: What are some typical regional issues to be voted on by the cities?

A: Establishment of a community college or an economic development district.

Q: If our team doesn't like the propositions presented to it on a ballot, do we have any alternatives?

A: Yes. First, reject the proposition presented to your team. Then, propose the launching of a study (feasibility, planning, economic, or engineering) to evaluate the merits of one or more possible alternative approaches to meeting the need--whatever your team wants to do. Put that proposed study on the ballot, as a write-in proposition. If it carries in the election (as it likely would do), it will be evaluated by the Game Management Center, and the results of that review will be supplied to your team during the next cycle, usually. Your team can then draft a firm ballot proposition (raising taxes, floating bonds, authorizing annexation, etc.) with the help of your Team Advisor, and the matter can be given to the GMC for inclusion on the ballot in the following cycle.

Q: What happens to a substitute proposition if the GMC questions its inclusion on a city election ballot? Is there any recourse?

A: Yes. Your team may refer the matter then and there to the regional team for consideration by all cities in the region. An example of such action might be the extension of utilities to serve a newly developing area which would be of benefit to other cities.

Q: What are some of the alternative user groups--other than those most especially interested in education--for which ELI might be helpful?

A: In its present form, or with slight modification of issues and ballot propositions, ELI could be used for leadership development, volunteer program services, studies in aging, public policy, economic problems analysis, community development, and group dynamics learning. Modifications to accommodate these are being studied for later simulations.

The game of ELI can be expanded in a number of directions, as indicated above. It can also be used in a variety of settings. Among these are as one-day training programs for decision-makers, for executives, and for managers-in-training. It can be used at meetings of professional associations, and at orientation sessions for public officials in schools, cities, counties, and other public bodies. By playing only one or two sessions every week, ELI can be an effective teaching tool to help students --graduate, undergraduate, or secondary--gain insight into the problems of balancing often conflicting demands from the community while steering a positive route into the unknown waters of the future. That is the spirit of ELI!

APPENDIX A.
SHORT GLOSSARY OF COMPUTER TERMS

ASCII	American Standard Code for Information Interchange
ANALYST	Person skilled in definition and development of techniques for solving problems
APPLICATION	The system or problem to which a computer is applied
ARTIFICIAL INTELLIGENCE	The study of computer and related techniques to supplement the intellectual capabilities of man
ASSEMBLE	To operate or perform the functions of an assembler
ASSEMBLER	A computer program which operates on symbolic data to produce executable computer instructions
ASYNCHRONOUS	Indicates that the execution of one operation is dependent on a signal that the previous operation has been completed
BASIC	Beginner's All-Purpose Symbolic Instruction Code (widely used)
BAUD	A unit of signalling speed equal to the number of code elements transmitted, expressed in bits per second (eg, 300 Baud)
BINARY	Refers to choice from between only two alternatives
BINARY SEARCH	Process of searching a list, dividing the list into two parts, until the required item is found (or not found)
BIT	Abbreviation for "Binary DigIT"
BOOLEAN	Refers to the use of "and," "or," and "not" in computer programming for making decisions in a logical flow
BOOTSTRAP ("BOOTING")	Technique for loading the first few instructions into a computer at start-up time
BUFFER	Intermediate storage area within a computer
BYTE	A measurable portion of consecutive binary digits (or BITS); usually an 8-BIT or a 6-BIT byte
CRT (CATHODE RAY TUBE)	A computer terminal
CHARACTER	A letter, number, or special symbol
CIRCUIT	A communication link between two or more points
COBOL	COmmon Business Oriented Language. A computer language widely-used in business applications

SHORT GLOSSARY (cont.)

CODE	A system of symbols for meaningful communication between people and computers
COLLATE	To merge two or more elements into an ordered set of data
COLUMN	The top-to-bottom positions on a page of data
COMMAND	A meaningful instruction (keyword) that gives direction to a computer system
COMMENT	See Documentation
COMPILER	A computer program that is more powerful than Assembler, and which translates computer language instructions into machine-executable code (used with higher level languages)
COMPUTER	A device capable of accepting information, applying prescribed processes to the information, and supplying results of those processes
CONDITIONAL	Based upon a decision made in the logic of a computer program
CONSTANT	Not subject to change
CONTROL	That part of a computer which determines the proper execution sequence
CONVERSION	The change of data from one form to another
CORE	Main memory of a computer
CP/M	Control Program for Microprocessors
CPU (CENTRAL PROCESSING UNIT)	The place where all the action of a computer takes place
CYCLE	A loop, an iteration
DATA	A general term to denote all facts, numbers, letters and symbols that refer to objects, ideas, situations, and conditions
DATA PROCESSING	Production of records and reports
DEBUG	To locate and correct errors in a computer program
DECISION	Computer operation to determine relationships between pieces of data within a computer program, and to select alternative courses of action
DIGIT	A sign or symbol to represent a number system

SHORT GLOSSARY (cont.)

DIGITAL	The use of an integral number system
DOCUMENTATION	Written text to describe "how" something is done or "how" it should be done
DOS	Disk Operating System
DUPLEX	Pertains to data being transmitted in both directions (to and from points)
EDIT	To arrange or rearrange data
EDP	Abbreviation for Electronic Data Processing
ERROR	Any deviation from a theoretically correct or true value
ERROR RATE	The total amount of information in error divided by the total amount of information received (a measure of circuit quality)
EXECUTE	To interpret a machine instruction
EXTERNAL MEMORY	The storage of data on a device which is not an integral part of the computer, such as tape, disk, or cards
FETCH	To obtain data from storage
FIELD	An assigned area within a record
FILE	An organized collection of related records
FIRMWARE	Programmed instructions implemented by the computer builder
FLOATING POINT	Calculation method which automatically accounts for location of the Radix Point (q.v.)
FLOWCHART	Graphic representation of the major steps in a program/system
FORTRAN	FORmula TRANslator. A programming language used in scientific and mathematical applications
GAME THEORY	A mathematical process developed by von Neumann for selecting an optimum strategy to respond to an opponent who has a strategy of his own
HARD COPY	A printout of data, a report
HARDWARE	The physical equipment of a computer
HEAD	A device which reads, records, or erases information on a storage device such as a disk drive or tape drive

SHORT GLOSSARY (cont.)

HSP	High Speed Printer
HSR	High Speed Reader
IDLE TIME	The time between the end of one program execution and the beginning of another
INPUT	Information which is transmitted from an external storage medium to the internal storage of a computer
INQUIRY	The interrogation of data from a computer keyboard
INSTRUCTION	An operation or command to be executed by the computer
INTERFACE	A common boundary between the components of a computer system
INTERPRETER	A computer execution routine that translates a program into machine code and executes the code
KEY	A group of characters used as an index to a larger body of information
LABEL	A set of symbols used to identify items, messages, or files, and can be an "address" in storage
LANGUAGE	A system used to represent and communicate information between people or between people and machines
LOGIC	The science dealing with reasoning and thought
LOOP	A set of instructions that is executed a given number of times in order to satisfy the conditions of a program
MACHINE LANGUAGE	The set of instructions (in the number system) that is basic to a computer system
MAGNETIC TAPE/DISK	Storage devices for a computer
MAINFRAME	Refers to a large computer system
MICRO COMPUTER	Refers to a small computer system (home variety)
MINI COMPUTER	Refers to a medium-sized system
MASTER FILE	A current file containing very pertinent information that can be up-dated periodically

SHORT GLOSSARY (cont.)

MATRIX	An array of quantities in a prescribed form of columns & rows
MERGE	To combine two or more elements into an ordered set of data
MODE	A computer system of representation (see Number System)
MODULE	An interchangeable plug-in item containing components
MONITOR	To supervise and verify correct operation of a program during its execution
MULTIPLEX	Process of transfering data from several storage devices operating at low speeds to one storage device operating at a high transfer speed
NUMBER SYSTEMS	Most commonly used in computer operations include these: Decimal - Base 10 Binary - Base 2 Octal - Base 8 Hexadecimal - Base 16
OBJECT CODE	The code that is the result of a compilation of a computer language such as COBOL, FORTRAN and Pascal
OPTIMUM CODE	A computer code that has been made efficient for execution
OUTPUT	Information transferred from the internal memory of a computer to external storage or to any device outside the computer, such as reports
OVERLAY	A technique for making larger computer programs into "pages," or blocks, for more efficient computer usage
PARALLEL	Simultaneous transmission of two or more parts
PARAMETER	A constant or variable upon which a computer program acts
PASCAL	A high-level computer language developed by Niklaus Wirth for teaching students how to program. Very structured. Named in honor of French mathematician Blaise Pascal
PARITY	A check to verify that data is being written and read correctly (either odd or even)
PERIPHERAL	Auxilliary devices other than the main part (CPU) of a computer
PORTABILITY	Ability to transfer programs and related data from one computer system to another
PRECISION	Degree of exactness

SHORT GLOSSARY (cont.)

PROGRAM	The complete plan for the solution of a problem
PSEUDO-INSTRUCTION	An instruction stated in ordinary English, which must be converted into a machine language before execution
QUEUING THEORY	A form of probability used to study delays or line-ups at service points
RADIX	Refers to Number Systems (q.v.)
RAM	Random Access Memory. Memory that can be accessed in a specified time without regard to its exact location
RANDOM NUMBER GENERATOR	Special routine for obtaining numbers according to specified limitations
RECORD	Group of related facts (fields) treated as a unit
ROW	The left-to-right arrangement of characters on a page
SEARCH	To look for specified data
SEGMENT	Overlay, or page of data (see Overlay)
SERIAL	Handling of data in a singular fashion (bit by bit)
SIMULATION	Representation of physical systems and phenomena by a computer
SOFTWARE	The totality of programs and routines used by a computer (compilers, assemblers, programs)
SORT	To arrange unorganized data into meaningful orders
UPDATE	The modification of information to reflect new data
VARIABLE	A quantity which can assume different values over a period of time
WORD	One storage location in a computer (usually consisting of one or two alphabetical characters)

APPENDIX B.
COMPUTER PSEUDO-CODE AND FOUR LANGUAGE APPLICATIONS

Pseudo-Code

The following is an example of computer instructions that might be written in Pseudo-Code. This example simulates the reading of player role data from a disk file, then moving it into appropriate "hold areas" within the program.

```
1)  OPEN PLAYER ROLE FILE
2)  READ PLAYER ROLE DATA
        IF END-OF-FILE GO TO 5)
3)  MOVE PLAYER DATA TO HOLD AREAS
4)  GO TO 2)
5)  CLOSE PLAYER ROLE FILE
6)  END
```

It is the programmer's task to code the above "English" onto the appropriate computer language. Following are four such examples using FORTRAN, BASIC, COBOL, and Pascal.

FORTRAN

```
    Dimension ID(12),ETH(12),EDUC(12),AG(12),INC(12)
 5  READ (6,10)PLID,ETHNIC,ED,AGE,INCOME
10  FORMAT(I2,I1,I1,I1,I1)
    IF(EOF(6))999,20
20  J=J+1
    ID(J)=PLID
    ETH(J)=ETHNIC
    EDUC(J)=ED
    AG(J)=AGE
    INC(J)=INCOME
    GO TO 5
999 END
```

BASIC

```
10  DIM ID(12),ETH(12),ED(12),AGE(12),INC(12)
20  OPEN"I",1,"A:PLAYER.DAT",6
30  IF EOF(1) THEN 80
40  INPUT #1,I,E1,E2,A,IN
50  SUB=SUB+1
60  ID(SUB)=I:ETH(SUB)=E1:ED(SUB)=E2
70  AGE(SUB)=A:INC(SUB)=IN:GOTO 30
80  CLOSE 1:END
```

(cont.)

BEST COPY AVAILABLE

COMPUTER PSEUDO-CODE AND FOUR LANGUAGE APPLICATION (cont.)

COBOL

```
WORKING-STORAGE SECTION.

Ø1  PLAYER-RECORD
    Ø3 ID          PIC 99.
    Ø3 ETHNIC      PIC 9
    Ø3 EDUCATION PIC 9.
    Ø3 AGE         PIC 9.
    Ø3 INCOME      PIC 9.

Ø1  TABLE-OF-PLAYERS.
    Ø3 PLAYER OCCURS 12 TIMES.
       Ø3 ID-X    PIC 99.
       Ø3 ETH-X   PIC 9.
       Ø3 ED-X    PIC 9.
       Ø3 AGE-X   PIC 9.
       Ø3 INC-X   PIC 9.

PROCEDURE DIVISION.
MAIN-LINE.
    OPEN PLAYER-FILE.
READ-FILE.
    READ PLAYER-FILE INTO PLAYER-RECORD
         AT END
             CLOSE PLAYER-FILE
             GO TO END-OF-JOB.
    COMPUTE I = I + 1.
    MOVE ID TO ID-X(I).
    MOVE ETHNIC TO ETH-X(I).
    MOVE EDUCATION TO ED-X(I).
    MOVE AGE TO AGE-X(I).
    MOVE INCOME TO INC-X(I).
    GO TO READ-FILE.

END-OF-JOB.
    STOP RUN.
```

Pascal

```
PROGRAM ELI;
TYPE
    PLAYER=RECORD
           ID:INTEGER;
           ETHNIC:INTEGER;
           ED:INTEGER;
           AGE:INTEGER;
           INCOME:INTEGER;
           END;
```

(cont.)

COMPUTER PSEUDO-CODE AND FOUR LANGUAGE APPLICATION (cont.)

Pascal (cont.)

```
    PLAY=FILE OF PLAYER;
VAR
    PLAYDATA:PLAY;
    PLAY_ARRAY:ARRAY[1..12] OF PLAYER;
    I:INTEGER;
BEGIN
    I:Ø;
    RESET(PLAYDATA, 'PLAYER.DATA');
    WHILE NOT EOF(PLAYDATA) DO
       BEGIN
         GET(PLAYDATA);
         IF NOT EOF(PLAYDATA) THEN
         BEGIN
            I:=I+1;
            PLAY_ARRAY[I].ID:=PLAYDATA↑.ID;
            PLAY_ARRAY[I].ETHNIC:=PLAYDATA↑.ETHNIC;
            PLAY_ARRAY[I].ED:=PLAYDATA↑.ED;
            PLAY_ARRAY[I].AGE:=PLAYDATA↑.AGE;
            PLAY_ARRAY[I].INCOME:=PLAYDATA↑.INCOME
         END
       END;
   CLOSE(PLAYDATA,LOCK)
END.
```

Note: Examples in other languages are not given here as higher level languages are easier to work with and better understood by most programmers.

BEST COPY AVAILABLE

SELECTED BIBLIOGRAPHY

Boocock, S.S., and E.O. Schild. Simulation Games in Learning. Beverly Hills: Sage Publications, 1968.

Coleman, Andrew M. Game Theory and Experimental Games. New York: Pergamon Press, 1982.

Coppard, Larry C., and Frederick Goodman. Urban Gaming/Simulation. Ann Arbor: University of Michigan, 1980.

Davis, Norton D. Game Theory. New York: Basic Books, 1970.

Duke, Richard D. Gaming: the Future's Language. New York: John Wiley and Sons, 1974.

Dukes, Richard L. Learning with Simulations and Games. Beverly Hills: Sage Publications, 1978.

Harrison, James D. Community and Environmental Simulations. Monticello (Ill.): Vance Bibliographies, 1981.

Hyman, Ronald T. Simulation Gaming for Values Education. New Brunswick: University Press of America, 1978.

Luce, R. Duncan, and Howard Raiffa. Games and Decisions. New York: John Wiley and Sons, 1957.

Megarry, Jacquetta. Aspects of Simulation and Gaming. London: Kogan Page, 1977.

Meier, Richard. Computer Simulation of a Community for Gaming. Denver: American Association for the Advancement of Science, 1961.

Shubik, Martin. The Uses and Methods of Gaming. New York: Elsevier Scientific Publishing Company, 1975.

Taylor, John L., and Rex Wlford. Learning and the Simulation Game. Milton Keynes: Drew University Press, 1978.

Twelker, Paul A., and Ken Lamden. Educational Simulation/Gaming. ERIC Clearinghouse on Media and Technology, 1972.

von Neumann, J. Theory of Games and Economic Behavior. Princeton: Princeton University Press, 1947.